Pluggable Neural Machine Translation Models via Memory-augmented Adapters

Yuzhuang Xu[1,†], Shuo Wang[1,†], Peng Li[2,*], Xuebo Liu[3]
Xiaolong Wang[1], Weidong Liu[1,4], Yang Liu[1,2,*]
[1]Department of Computer Science & Technology, Tsinghua University, Beijing, China
[2]Institute for AI Industry Research (AIR), Tsinghua University, Beijing, China
[3]Harbin Institute of Technology, Shenzhen, China
[4]Zhongguancun Laboratory, Beijing, China
{xyz21thu,wangshuo.thu}@gmail.com, lipeng@air.tsinghua.edu.cn
liuyang2011@tsinghua.edu.cn

Abstract

Although neural machine translation (NMT) models perform well in the general domain, it remains rather challenging to control their generation behavior to satisfy the requirement of different users. Given the expensive training cost and the data scarcity challenge of learning a new model from scratch for each user requirement, we propose a memory-augmented adapter to steer pretrained NMT models in a pluggable manner. Specifically, we construct a multi-granular memory based on the user-provided text samples and propose a new adapter architecture to combine the model representations and the retrieved results. We also propose a training strategy using memory dropout to reduce spurious dependencies between the NMT model and the memory. We validate our approach on both style- and domain-specific experiments and the results indicate that our method can outperform several representative pluggable baselines. Code and data are available at https://github.com/xuyuzhuang11/StyleMT

Keywords: Neural machine translation, style / domain customization, pluggable, memory, adapter.

1. Introduction

In recent years, modern neural machine translation (NMT; Vaswani et al., 2017) systems are often developed with large-scale parallel data extracted from the Web (Liu et al., 2020; Fan et al., 2021), whose style and content are driven by the average distribution of data from many domains (Vu and Moschitti, 2021). Therefore, the performance of strong NMT models is close to or even better than human translators in the general domain (Hassan et al., 2018; Kocmi et al., 2022).

However, MT customers may have some special requirements, including both style- and domain-specific individual demands (Michel and Neubig, 2018; Zhang et al., 2022). For instance, some users may want translations in a special style, while some others may need to translate medical texts. These requirements can be quite diverse among different customers and retraining or fine-tuning the model for each user entails significant development costs, especially with limited data from users.

Fortunately, *pluggable* methods (Keskar et al., 2019; Dathathri et al., 2020; He et al., 2021a) bring hope to handle the aforementioned user requirements, which employ additional modules to steer pretrained models. As shown in Figure 1, the users can provide some text samples for the NMT model to imitate. We will then learn a plugin to control the NMT model to satisfy the user demands without optimizing the parameters in the original model, by which we can maintain the performance of the pretrained model, alleviating the risk of catastrophic forgetting (Kirkpatrick et al., 2017).

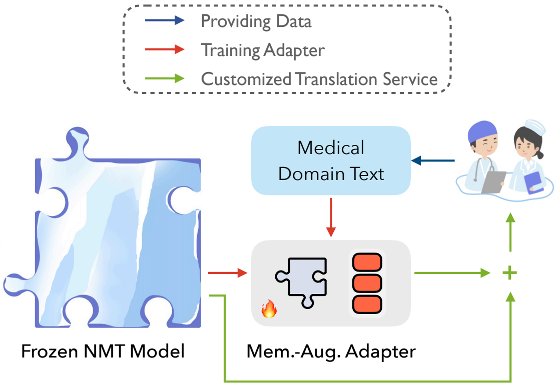


Figure 1: A frozen and pluggable NMT model using memory-augmented plugins. For each user group with special requirements, we can develop a plugin for them without affecting other users.

Some researchers suggest lightweight parametric plugins for controlling pretrained models (Houlsby et al., 2019; Bapna and Firat, 2019; Pfeiffer et al., 2021; Rücklé et al., 2021; Li and Liang, 2021; Mao et al., 2022). For machine translation, such plugins can also tailor model behavior for diverse user demands. However, recent studies find that there exists a performance bottleneck

[†]Equal contribution
[*]Corresponding authors

of fully-parametric pluggable methods (Bapna and Firat, 2019; Li and Liang, 2021; Ding et al., 2022): increasing the number of trainable parameters can not always lead to better performance. Inspired by the recent progress of retrieval-augmented models (Lewis et al., 2020; Khandelwal et al., 2020, 2021; He et al., 2021a,b), we propose to increase the expressive power (Li and Liang, 2021) of parametric plugins through external memories, which we term it *memory-augmented adapter*.

The main challenges of the memory-augmented adapter are two-fold: (1) *constructing* memories that can provide useful customization information; and (2) *integrating* the memories into existing NMT models without quality loss. Although long phrases can provide more contextualized information, matching long sequences between queries and memory items is more difficult than matching shorter ones. We propose to build multi-granular memories to balance the amount of contextualized information and the retrieval difficulty. Unlike many previous works (Khandelwal et al., 2020, 2021; He et al., 2021a) that encode the source sentence and the target prefix as the key and the next token as the value, our memory can provide multi-scale translation knowledge (Li et al., 2022) that is suitable for queries coming from different layers of the NMT model (Hewitt and Liang, 2019). For memory integration, we propose a *new adapter architecture* to better interpolate the original model representation and the retrieved vectors. Moreover, we propose a new training strategy with memory dropout to reduce spurious dependencies between the NMT model and the provided memory. We conduct experiments for both style- and domain-related customizations and the results show the superiority of our method over many representative baselines.

2. Related Work

2.1. Style / Domain Adaptation for NMT

Adapting NMT models to specific style or domain texts has been investigated in several previous works (Luong and Manning, 2015; Niu et al., 2017; Chu and Wang, 2018). For stylized NMT, many previous works focus on the formality control of translations (Niu et al., 2018; Wu et al., 2021b), of which the style has a clear definition. Most existing works need to train a specific model for each style. For instance, Niu and Carpuat (2020) mix the training data of both style transfer and machine translation to learn a formality-sensitive NMT model. Given that the user-provided styles can be of great diversity, we aim to satisfy different style demands in a pluggable manner.

For domain adaptation, Luong and Manning (2015) propose an effective method that fine-tunes an out-of-domain model with small-sized in-domain supervised corpora. Hu et al. (2019) further design an unsupervised method, since parallel data is hardly available in many domains. Zheng et al. (2021) extend kNN-MT (Khandelwal et al., 2021) to perform unsupervised domain adaptation. Our work is different from Zheng et al. (2021) in both memory design and usage and the experiments show that our proposed framework performs better than their approach.

2.2. Machine Translation Customization

Machine translation customization aims to satisfy the special requirements of different users. Vu and Moschitti (2021) propose to select data that is similar to the user-provided text samples and then train or fine-tune an NMT model for the corresponding user. Following Michel and Neubig (2018), we believe that MT customization has some specific traits that distinguish it from common style and domain adaptation settings: (1) The number of customization requirements is very large due to the personal variation among different MT system users; (2) The available data is often very limited (even monolingual, let alone parallel) for each customization requirement. Thus, we propose to leverage pluggable methods to customize existing NMT models.

2.3. Pluggable Pretrained Models

Pluggable methods aim to control the generation behavior of pretrained models without optimizing model parameters (Dathathri et al., 2020; Yang and Klein, 2021; Liu et al., 2021). Some works propose to use parametric plugins, with Bapna and Firat (2019) and Houlsby et al. (2019) inserting some adapters, Li and Liang (2021) prepending some trainable vectors, and Hu et al. (2022a) leveraging low-rank decomposition of matrices. Retrieval-augmented models, also treated as pluggable methods, augment the model with non-parametric memory. kNN-MT (Khandelwal et al., 2021) combines the model prediction and retrieval distribution at the output layer. Borgeaud et al. (2022) build a chunk-level memory for language modeling. Chen et al. (2022) encode questions and answers into key-value pairs for question answering. In this work, we aim to combine the merits of both parametric and non-parametric plugins and propose a new type of memory for NMT, which explicitly considers the phrases of different granularities.

3. Background

3.1. Transformer Model

We first give a description of some components in the Transformer (Vaswani et al., 2017) model.

Given the input sentence $\mathbf{x}$, Transformer maps it into vectors via an encoder:

$$\mathbf{E} = \mathrm{encoder}(\mathbf{x}) \tag{1}$$

where $\mathbf{E} \in \mathbb{R}^{|\mathbf{x}| \times d}$ and d is the hidden size of the model. The encoder output is then utilized by the decoder, which is a stack of several independent layers. We use $\mathbf{D}^{(i)}$ to denote the output of the i-th decoder layer. Specifically, each decoder layer firstly employs a self-attention module to model the dependency between the target-side words:

$$\begin{aligned} \mathbf{S}^{(i)} &= \mathrm{attn}(\, \mathbf{D}^{(i-1)}, \mathbf{D}^{(i-1)}, \mathbf{D}^{(i-1)} \,) \\ \mathbf{L}_1^{(i)} &= \mathrm{layernorm}(\, \mathbf{D}^{(i-1)} + \mathbf{S}^{(i)} \,) \end{aligned} \tag{2}$$

where $\mathrm{attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ is the multi-head attention and $\mathrm{layernorm}$ is the layer normalization.

After that, a cross-attention module is adopted to integrate the source-side information:

$$\begin{aligned} \mathbf{C}^{(i)} &= \mathrm{attn}(\, \mathbf{L}_1^{(i)}, \mathbf{E}, \mathbf{E} \,) \\ \mathbf{L}_2^{(i)} &= \mathrm{layernorm}(\, \mathbf{L}_1^{(i)} + \mathbf{C}^{(i)} \,) \end{aligned} \tag{3}$$

The output of the cross-attention module is then projected with a feed-forward layer. The decoder output is finally used to estimate the probability $P(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta})$, where $\mathbf{y}$ is the target sentence and $\boldsymbol{\theta}$ denotes the set of model parameters.

3.2. Style Customization in NMT

Similar to generating images with specific styles (Jing et al., 2022; Ruiz et al., 2023), style customization in NMT means outputting translations with user-specified styles (Michel and Neubig, 2018; Syed et al., 2020). For example, we want to generate translations in Shakespeare style in a Zh-En translation task. A simple example is as follows.

Zh: 哦上帝啊，请赐予我力量吧！
En (G): Oh God, please grant me strength!
En (S): Oh Lord, do thou endow me with thy might!

"**En (G)**" denotes the output of vanilla translation model, and "**En (S)**" denotes the output of style-customized translation model using Shakespeare corpus. The underlined expressions are typical representations of Shakespeare style.

The task most closely related to style customization in NMT is author-stylized rewriting (Syed et al., 2020; Singh et al., 2021), which aims to rewrite a given text in the style of a specific author. Syed et al. (2020) sum up the user or author style into three levels, namely surface level, lexical level, and syntactic level styles (Syed et al., 2020). These levels capture subtle differences in punctuation, word usage, and even sentence construction unique to individual authors, thereby making author-stylized rewriting a challenging task. Style customization in NMT not only shares the same challenges as author-stylized rewriting, but it must also simultaneously translate the provided text into the target language, presenting its own unique challenges.

4. Approach

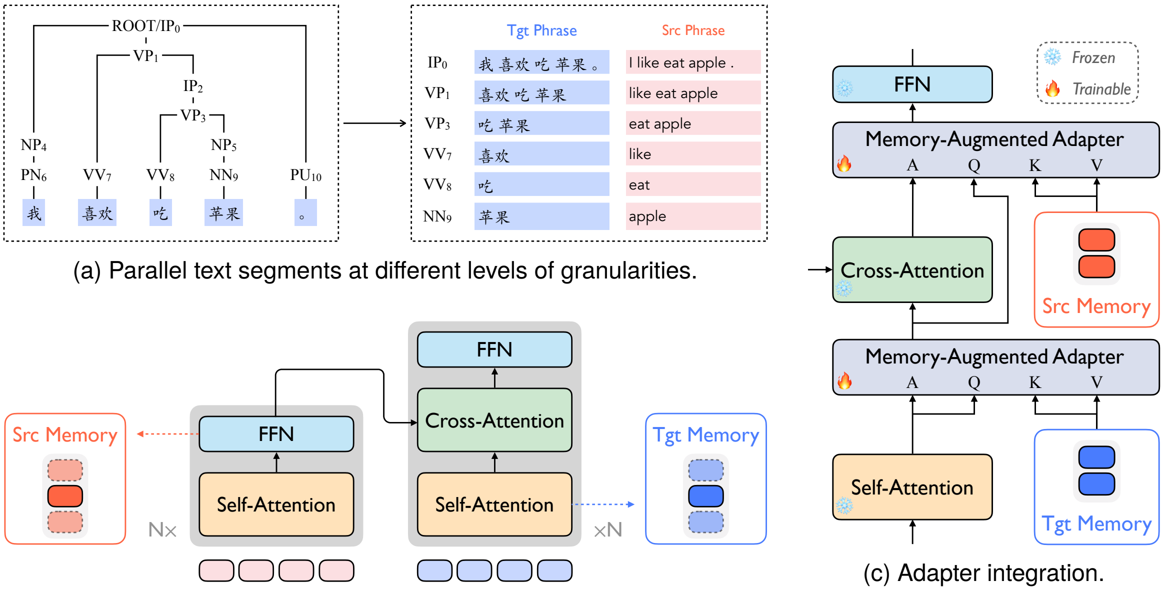
4.1. Overview

In this work, we aim to enable NMT users to control existing NMT models by simply providing some examples. To this end, we propose a memory-augmented adapter to help NMT models imitate the user-provided text samples. Specifically, we propose the multi-granular memory that can better leverage multi-scale patterns, which have proven to be important for NMT (Li et al., 2022). We also propose a new adapter (i.e., memory-augmented adapter) to integrate external memory into NMT models. We will explain how to construct and utilize the memory in the following two subsections.

4.2. Multi-granular Continuous Memory

Our memory needs not only to extract essential information from user-provided text but also to be easy to retrieve for models. For the first objective, we propose to build the memory with parallel phrase pairs, which reflect the translation pattern required by the customer. However, it is non-trivial to determine the granularity of the used phrases. Storing only short ones may waste a lot of contextualized information while storing too many long phrases would make it rocky to match the query and the memory items. To address this issue, we propose to construct a multi-granular memory to balance the amount of contextualized information and the retrieval difficulty. As shown in Figure 2a, we use parse trees to extract multi-granular phrases, which can identify more meaningful boundaries than random splitting. The extracted phrases are then translated by NMT models to form parallel phrase pairs.

For the second objective, we propose to use the same model to build and utilize the memory. For each phrase pair, we perform forward computation to get the continuous representation at each layer of the involved NMT model. We store the encoder output $\mathbf{E}$ as the source-side memory and the self-attention output $\mathbf{S}^{(i)}$ at every decoder layer as the target-side memory. See Eq. (1) and (2) for more details of the stored representations. Each memory item is averaged among the representations of all tokens in a phrase, whose size is d. Figure 2b shows an example. The reason we extract $\mathbf{E}$ and $\mathbf{S}^{(i)}$ as our memory is that these representations are at the same layer where we perform memory



(a) Parallel text segments at different levels of granularities.

(b) Construction of multi-granular continuous memory.

(c) Adapter integration.

Figure 2: Construct and integrate memories. (a) We leverage parse trees to obtain multi-granular phrases. Each monolingual phrase is then translated by NMT models. (b) For each phrase pair, we perform a forward computation in the teacher-forcing manner and record some intermediate representations into the memory. (c) Illustration of adapter integration. The adapter retrieve and leverage the memories.

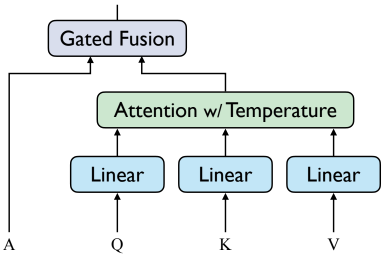


Figure 3: Memory-augmented adapter architecture.

retrieval. Our motivation is to narrow down the gap between the memory items and the queries, making it easier for the model to read the memory.

We focus on using monolingual user-provided data in this work since parallel data is often unavailable for most requirements. However, our method can be easily extended for bilingual data, from which we can automatically extract phrase pairs based on unsupervised word alignment algorithms (Dyer et al., 2013; Chen et al., 2021).

4.3. Memory-augmented Adapter

Adapter Architecture We propose a new type of adapter to read memory. The memory-augmented adapter has 4 inputs: anchor, query, key, and value, which can be represented as $\mathbf{A}$, $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$. Anchors and queries are derived from the frozen NMT model while keys and values come from the memory. As depicted in Figure 3, we use an attention module to generate the retrieved result:

$$\mathbf{R} = \mathrm{softmax}(\, \mathbf{Q}\mathbf{W}_q\mathbf{W}_k^\top\mathbf{K}^\top / \, T\,)\mathbf{V}\mathbf{W}_v \qquad (4)$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d\times d}$ and the retrieved result $\mathbf{R}$ has the same shape with $\mathbf{Q}$. T is a hyper-parameter to control the sharpness of the retrieval distribution. To avoid the model being completely dependent on the retrieved result $\mathbf{R}$ that can be erroneous in some cases, we also take in an anchor from the original model, which is combined with $\mathbf{R}$ via a gated fusion module:

$$\begin{aligned} \lambda &= \mathrm{sigmoid}(\, \mathrm{relu}(\, [\mathbf{A};\mathbf{R}]\mathbf{W}_1\,)\mathbf{W}_2\,) \\ \mathbf{O} &= \lambda\,\mathbf{A} + (1-\lambda)\,\mathbf{R} \end{aligned} \qquad (5)$$

where $\mathbf{O}$ is the adapter output, which has the same shape as the anchor $\mathbf{A}$. $\mathbf{W}_1 \in \mathbb{R}^{2d\times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{d\times 1}$. λ is the learned interpolation ratio.

Adapter Integration We apply the memory-augmented adapter to the self- and cross-attention modules in the decoder, since these two components are important for target-side language modeling and source-side information utilization. At the i-th decoder layer, we use the self-attention output $\mathbf{S}^{(i)}$ as queries to read the target-side memory:

$$\mathbf{O}_1^{(i)} = \mathrm{memadapt}(\, \mathbf{S}^{(i)}, \mathbf{S}^{(i)}, \mathbf{M}_t^{(i)}, \mathbf{M}_t^{(i)}\,) \qquad (6)$$

where $\mathrm{memadapt}(\mathbf{A}, \mathbf{Q}, \mathbf{K}, \mathbf{V})$ denotes the memory-augmented adapter. $\mathbf{M}_t^{(i)} \in \mathbb{R}^{N_t^{(i)}\times d}$ represents the target-side memory, where $N_t^{(i)}$

denotes the number of items in $\mathbf{M}_t^{(i)}$. The adapter output $\mathbf{O}_1$ is then provided to the layer normalization module:

$$\mathbf{L}_1^{(i)} = \text{layernorm}(\, \mathbf{D}^{(i-1)} + \mathbf{O}_1^{(i)}\,) \qquad (7)$$

Similarly, we read the source-side memory in the cross-attention module:

$$\begin{aligned} \mathbf{O}_2^{(i)} &= \text{memadapt}(\, \mathbf{C}^{(i)}, \mathbf{L}_1^{(i)}, \mathbf{M}_s^{(i)}, \mathbf{M}_s^{(i)}\,) \\ \mathbf{L}_2^{(i)} &= \text{layernorm}(\, \mathbf{L}_1^{(i)} + \mathbf{O}_2^{(i)}\,) \end{aligned} \qquad (8)$$

Figure 2c shows an example. To reduce the redundancy that a phrase pair would repeatedly appear in memories at every decoder layer, we split all the phrase pairs into L parts, where L is the number of decoder layers. Each layer only stores one part of phrase pairs.

Training Strategy Inspired by dropout (Srivastava et al., 2014) that can effectively reduce spurious co-adaptation between model parameters, we propose a memory dropout approach to prevent NMT models from being too dependent on some specific memory items. When training the memory-augmented adapter, we randomly drop part of the memory items. Let $\mathbf{M}$ be the full memory and $\hat{\mathbf{M}}$ be the remained memory after memory dropout, the overall loss can be given by

$$\begin{aligned} \mathcal{L} = \; & \underbrace{\mathcal{L}_{\text{NLL}}(\, P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}, \mathbf{M})\,)}_{\text{loss using full memory}} \\ + \; & \underbrace{\alpha\, \mathcal{L}_{\text{NLL}}(\, P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{M}})\,)}_{\text{loss using dropped memory}} \\ + \; & \underbrace{\beta\, \mathcal{L}_{\text{dist}}(\, P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}, \mathbf{M}), P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{M}})\,)}_{\text{loss modeling the agreement}} \end{aligned} \qquad (9)$$

where α and β are hyperparameters. $\mathcal{L}_{\text{NLL}}$ is the conventional negative log-likelihood. The agreement loss (i.e., $\mathcal{L}_{\text{dist}}$) (Kambhatla et al., 2022) measures the distance between two distributions:

$$\mathcal{L}_{\text{dist}}(p, q) = \frac{1}{2}(D_{\text{KL}}(p||q) + D_{\text{KL}}(q||p)) \qquad (10)$$

Extension Since our method does not change the model decoding, it can also be combined with the retrieval-based decoding algorithm as shown in kNN-MT (Khandelwal et al., 2021), which interpolates the model probability with a retrieved distribution. We call this decoding method kNN decoding.

5. Style Customization

5.1. Setup

NMT Model Training In the pluggable scenario, we should first have an existing NMT model, which can serve as the foundation for further customization. We use the training corpus of the WMT20 En↔Zh translation task[1] to train NMT models, which contains 23.9M sentence pairs. We use `SentencePiece`[2] to preprocess the data and the sentence-piece model we used is released by mBART (Liu et al., 2020). The architecture of our NMT models is Transformer (Vaswani et al., 2017), whose hidden size is 512 and depth is 6. Please refer to Section 10.1 in appendix for more details.

Customization Data We evaluate the customization effect of our method in two translation directions: En→Zh and Zh→En. We use the works of two world-renowned writers as stylized text samples, including Shakespeare and Lu Xun. Their works created representative styles for English and Chinese, respectively. We extract their texts from the web and then split the data into training, validation, and test sets. For Shakespeare's style, the training set contains 20K English sentences while the validation and test sets contain 500 sentences, respectively. The target-language (i.e., English) training and validation sentences are then translated by the NMT model, while the test set is translated by human translators. For Lu Xun's style, the training set consists of 37K sentences while the validation and test sets contain 500 sentences. Similarly, the test set is also translated by humans while the training and validation sets are translated by NMT models. The resulting corpus is called **M**achine **T**ranslation with **S**tyle **C**ustomization (MTSC). We will add more styles in different languages for MT research in the future.

Memory Construction We first build parse trees for target-side sentences using `Stanford Parser`[3] and then extract multi-granular phrases. As mentioned in Section 4.3, we evenly divide the extracted phrases according to their lengths into L parts to avoid information redundancy between different layers. We did not store the representations of phrases longer than a pre-specified threshold l_{max}, since the occurrence of long phrases is very low. l_{max} is set to 10 for Zh and 8 for En.

Adapter Training The general NMT model is frozen when training the memory-augmented adapter. We determine the value of the hyperparameters based on the validation performance. Specifically, the temperature in Eq. (4) is set to 0.5. Both the α and β in Eq. (9) are set to 5. The memory dropout rate is set to 0.1. We provide more details of adapter training in Section 10.1 in appendix.

[1] https://www.statmt.org/wmt20/translation-task.html
[2] https://github.com/google/sentencepiece
[3] https://nlp.stanford.edu/software/lex-parser.html

Table 1: Automatic evaluation for style customization. We highlight the **best** and second best scores.

Method	BLEU(↑)			Perplexity(↓)			Classifier Score(↑)		
	En-Zh	Zh-En	Avg.	En-Zh	Zh-En	Avg.	En-Zh	Zh-En	Avg.
Vanilla (Vaswani et al., 2017)	13.7	15.7	14.7	459.6	127.4	293.5	18.0	28.4	23.2
Extreme (Michel and Neubig, 2018)	16.0	17.7	16.9	315.2	113.7	214.5	37.4	43.4	40.4
Adapter (Houlsby et al., 2019)	16.8	19.4	18.1	351.1	121.0	236.1	33.8	58.2	46.0
MT+Rewrite (Syed et al., 2020)	16.3	15.7	16.0	222.4	127.5	349.8	47.0	28.4	37.7
kNN-MT (Khandelwal et al., 2021)	18.9	20.0	19.5	230.7	98.5	164.6	42.2	70.4	56.3
DExperts (Liu et al., 2021)	13.8	15.9	14.9	467.0	127.3	297.2	18.4	31.4	24.9
ChatGPT (OpenAI, 2022)	20.0	13.6	16.8	620.0	131.8	375.9	41.4	24.6	33.0
Memory-augmented Adapter	20.8	21.1	21.0	257.8	110.9	184.3	47.0	69.4	58.2
+ kNN decoding	**21.3**	**21.8**	**21.6**	**199.6**	**95.1**	**147.4**	**53.2**	**85.2**	**69.2**

Table 2: Human evaluation for style customization in En→Zh. The comparison is performed between kNN-MT and our method. "Win" means our method performs better. κ denotes Fleiss' kappa.

Human	Content Preservation			Sentence Fluency			Style Similarity		
	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose
Rator 1	64.0%	12.0%	24.0%	61.0%	9.0%	30.0%	69.0%	5.0%	26.0%
Rator 2	64.0%	13.0%	23.0%	57.0%	13.0%	30.0%	63.0%	11.0%	26.0%
Rator 3	67.0%	9.0%	24.0%	62.0%	5.0%	33.0%	71.0%	6.0%	23.0%
Avg.	65.0%	11.3%	23.7%	60.0%	9.0%	31.0%	67.7%	7.3%	25.0%
κ	0.476			0.446			0.656		

Baselines We compare our method with the following representative baselines: *Extreme* (Michel and Neubig, 2018), *Adapter* (Houlsby et al., 2019), *MT+Rewrite* (Syed et al., 2020), *kNN-MT* (Khandelwal et al., 2021), *DExperts* (Liu et al., 2021), *ChatGPT* (GPT3.5-turbo-0301; OpenAI, 2022).

Evaluation Metrics We use both automatic and human evaluation to thoroughly compare them. The automatic evaluation metrics are as follows:

- *BLEU*: measuring the translation quality of model outputs. We use sacreBLEU[4] (Post, 2018) to estimate the BLEU score.
- *Perplexity*: measuring the fluency of model outputs. We fine-tune a pretrained Transformer LM (Dai et al., 2019) with stylized text to calculate perplexity.
- *Classifier Score*: measuring the similarity between model outputs and the stylized text samples. We follow Li et al. (2018) to train style classifiers to quantify the style similarity. The classifier we used is TextCNN (Kim, 2014). It can achieve an accuracy of 93.5% for Lu Xun's style and 94.5% for Shakespeare's style. We use these classifiers to estimate whether the output is in the desired style.

5.2. Main Results

Automatic Evaluation Table 1 shows the performance of all the involved methods in the style customization task. When decoding with vanilla beam search, our method can outperform all the baselines in terms of BLEU and classifier score on average, indicating the effectiveness of the proposed memory-augmented adapter in controlling the output style of NMT models. The perplexity of kNN-MT is better than ours, but its BLEU score is much worse. Although ChatGPT customizes the translation at a small cost, the result is not satisfactory. When combined with kNN decoding, which is illustrated in **Extension** in Section 4.3, our method can be further improved, achieving the best performance across all the three automatic metrics. These results re-demonstrate that our method is complementary to kNN-MT.

Human Evaluation We also perform a human evaluation to assess the translation quality of different methods. We follow previous works (Zhang et al., 2018; Ke et al., 2019) to ask human evaluators to compare the outputs of different methods. Since human evaluation is time-consuming and labor-intensive, we only compare our method with the strongest baseline (i.e., kNN-MT) in En→Zh. Note that our outputs used for human evaluation are generated using vanilla beam search. Following Hu et al. (2022b), each sentence is evaluated in terms of content preservation, sentence fluency, and style similarity. Table 2 shows the results, from which we

[4] English-Chinese: nrefs:1 | case:mixed | eff:no | tok:zh | smooth:exp | version:2.3.1. Chinese-English: nrefs:1 | case:mixed | eff:no | tok:13a | smooth:exp | version:2.3.1.

Table 3: BLEU scores in the domain customization task. We highlight the **best** and <u>second best</u> scores.

Method	IT		Medical		Law		Koran		Avg.	
	valid	test	valid	test	valid	test	valid	test	valid	test
Vanilla	28.1	28.4	26.4	27.6	36.2	35.9	10.9	11.5	25.4	25.9
Adapter	<u>30.9</u>	30.5	26.8	28.0	36.0	35.6	12.9	13.5	26.7	26.9
kNN-MT	28.8	29.2	<u>30.0</u>	<u>32.3</u>	<u>38.3</u>	<u>38.4</u>	14.6	15.1	27.9	28.8
Memory-augmented Adapter	**31.2**	<u>31.1</u>	<u>30.0</u>	32.0	37.5	37.3	<u>14.7</u>	<u>15.3</u>	<u>28.4</u>	<u>28.9</u>
+ kNN decoding	30.5	**31.4**	**31.3**	**33.5**	**38.8**	**38.6**	**15.7**	**16.5**	**29.1**	**30.0**

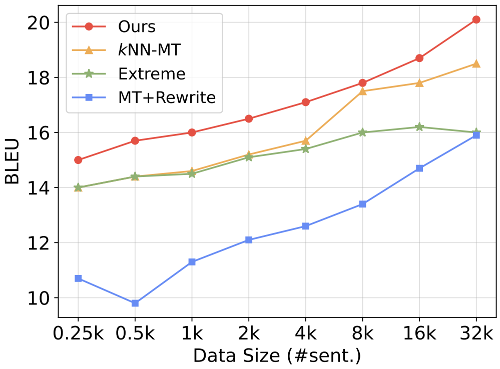


Figure 4: Performance of style customization at different data scales. "Ours" does not use kNN decoding.

find our approach performs better than the baseline in all the three evaluation aspects. The agreement of the three human evaluators is estimated through Fleiss' kappa (Fleiss, 1971) and the results demonstrate *moderate agreement* $(0.4 \leq \kappa \leq 0.6)$ in terms of both content preservation and sentence fluency and *good agreement* $(0.6 \leq \kappa \leq 0.8)$ regarding style similarity.

5.3. Performance at Different Data Scales

In some cases, the user-provided data can be of extremely small scale (Michel and Neubig, 2018). We thus investigate the performance of the involved methods using customization data of different scales. Figure 4 shows the results. Our memory-augmented adapter consistently outperforms the baselines at different data scales, even with only 250 exemplary sentences. These results show that our method can be applied to extremely low-resource adaptation scenarios.

6. Domain Customization

6.1. Setup

NMT Model Training We train the NMT model using the WMT14 De-En training corpus[5], including 4.5M sentence pairs. The training data is preprocessed in the same way as style customization.

Customization Data To evaluate the performance in the domain customization setting, we follow previous works (Aharoni and Goldberg, 2020; Zheng et al., 2021) to use a multi-domain dataset, which includes four domains: *IT*, *Medical*, *Law* and *Koran*. To simulate real-world user customization where the user-provided data is often of small scale, we randomly select 20K sentences for IT, Medical, and Law, and use all the 18K sentences for Koran. We also use only the target-side training data to simulate real-world cases and use NMT models to generate synthetic parallel data. All the validation and test sets are authentic parallel data.

Memory Construction and Adapter Training We filter phrases longer than 10 during memory construction. For adapter training, T is set to 0.1 for Medical and Law, and 0.5 for the other two domains. Both α and β are set to 5 on all the four domains. The memory dropout rate is set to 0.1.

Baselines We compare our proposed method with two representative pluggable domain adaptation baselines: adapter (Houlsby et al., 2019) and kNN-MT (Khandelwal et al., 2021).

6.2. Main Results

The adaptation performance on different domains is shown in Table 3. On average, our method can outperform the two baselines even without kNN decoding, demonstrating the effectiveness of our motivation to boost parametric plugins with external memory. When combined with kNN decoding, our method can achieve better results on Medical, Law, and Koran. Using kNN decoding, our method can improve 3.1 and 1.2 BLEU scores over Adapter and kNN-MT on the test sets, respectively.

6.3. Inference Time

A concern for retrieval-augmented methods is that they may significantly slow down the inference process. As shown in Figure 5, our method is slower than Adapter, but the difference between the two methods becomes very slight when using big batch sizes. For instance, our inference time is only 1.15 times that of Adapter with a batch size of 128. Our

[5] https://www.statmt.org/wmt14/translation-task.html

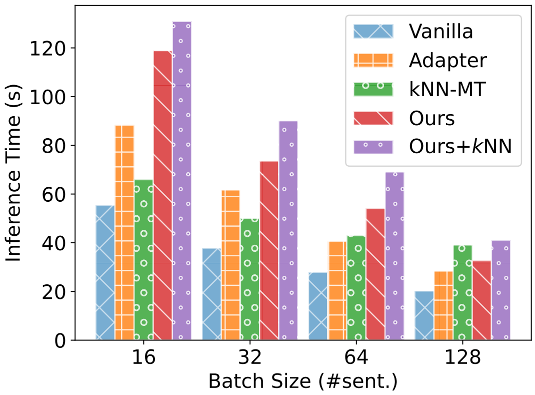


Figure 5: Inference time reported on the IT domain. “Ours” is not combined with kNN decoding.

method is also comparable to kNN-MT. In particular, when the batch size is set to 128, our method is slightly faster than kNN-MT. When also using kNN decoding, our method is slightly slower than kNN-MT (i.e., 41.1s vs. 39.1s with batch size = 128). We implement the kNN algorithm with `Faiss-gpu` (Johnson et al., 2017) to accelerate the retrieval process.

7. Discussion

7.1. Effect of Different Components

We conduct thorough ablation studies to better understand the effect of the proposed components.

Granularity Distribution As mentioned in Section 4.2, we divide all the phrase pairs into several different parts to reduce the redundancy of information among the decoder layers. Our basic idea is that a certain phrase pair only needs to appear in one layer of the decoder. The phrase pairs are divided according to their lengths. We investigate three ways to distribute the phrase pairs to the decoder layers: (1) *long-to-short* where the phrase length decreases from the bottom layer to the top layer; (2) *short-to-long* where the phrase length increases from the bottom layer to the top layer; and (3) *random* where the memory item of a certain phrase pair is stored by a randomly selected layer. Figure 6 shows the results of the three ways, where we find short-to-long achieves the best performance. We think the reason is that different layers may carry various types of linguistic properties in the Transformer model (Voita et al., 2019), which requires information of different granularities. When reading the memory, queries from lower layers may contain less contextualized information (Hewitt and Liang, 2019), thus short phrases are more suitable for them. At higher layers, long phrases that can provide more contextualized information performs better. We thus use short-to-long as the default setting.

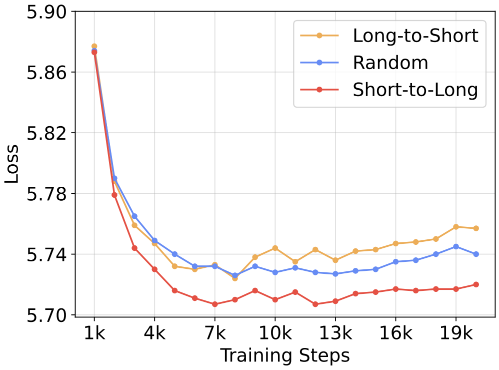


Figure 6: Effect of granularity distribution across the decoder layers. Each curve denotes the validation loss in the En-Zh style customization task. We only use the vanilla NLL loss to rule out the effect of the training strategy. “Long-to-Short”: lower layers store the representations of longer phrases while higher layers store the representations of shorter layers. “Random”: each phrase pair is stored in a randomly selected layer. “Short-to-Long”: lower layers store shorter phrases while higher layers store longer phrases.

Table 4: BLEU scores when using different memory dropout. The results are reported on the validation set in the En-Zh style customization task.

Method	BLEU
No memory dropout	18.2
+ item-level memory dropout	18.2
+ layer-level memory dropout	18.6

Effect of Memory Dropout We investigate the performance of two types of memory dropout: (1) *item-level memory dropout* that drop each item with a certain probability; and (2) *layer-level memory dropout* that drop all the memories at a decoder layer with a certain probability. Table 4 shows the results, where all the models are trained using the overall loss function (i.e., $\mathcal{L}$ in Eq. (9)). We find the layer-level memory dropout performs better. Therefore, we use layer-level memory dropout by default.

Effect of Memory Granularity To validate the necessity of building memory in a multi-granular form, we compare the performance of single- and multi-granular memories. Figure 7 shows the results, where we find using multi-granular memory can achieve lower validation loss, indicating the effectiveness of our method. We use multi-granular memory in other experiments by default.

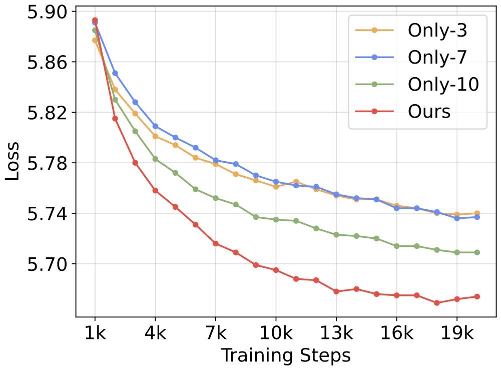


Figure 7: Comparison between single-granular and multi-granular. Each curve is plotted on the validation set in the En→Zh style customization task. “Only-x”: only using phrases of length x to build the memory. “Ours”: using the multi-granular memory.

Table 5: Analysis of memory usage.

Method	BLEU
Ours	18.6
– gated fusion	18.3
– source-side memory	16.5
– target-side memory	16.1

Analysis of Memory Usage Table 5 shows the effect of different components that are related to memory usage. Firstly, we notice that the gated fusion mechanism has a positive effect on translation quality, indicating the necessity of learning an input-dependent interpolation ratio between original model representations and retrieved results. In addition, we observe that there is a significant performance drop when using only either the source- or target-side memory. These results demonstrate that building parallel memory using phrase pairs is very useful.

Effect of Integration Layers We also integrate the memory into different layers to better understand our method. Table 6 shows the results, from which we find using the memories at all layers performs best and higher layers tend to be more important than lower layers. A potential reason is that memories at higher layers contain more contextualized information.

7.2. Comparison with Fine-tuning

Although our goal in this work is to better build pluggable NMT models, our method is not only limited to this setting. For instance, the proposed memory-augmented adapter can also be used when the NMT model is not frozen (i.e., fine-tuning (Luong and Manning, 2015)). Table 7 shows the results, where we observe that our method can also improve the performance of fine-tuning. This result implies that the external memory may provide essential information that is complementary to that stored in model parameters.

Table 6: Model performance when integrating memory into different layers. "✓": equipped with the memory. "-": not equipped with the memory.

Selected Layers						BLEU
L1	L2	L3	L4	L5	L6	
✓	-	-	-	-	-	15.1
✓	✓	-	-	-	-	15.5
✓	✓	✓	-	-	-	15.6
✓	✓	✓	✓	-	-	16.2
✓	✓	✓	✓	✓	-	16.9
✓	✓	✓	✓	✓	✓	**18.6**
-	✓	✓	✓	✓	✓	17.5
-	-	✓	✓	✓	✓	17.2
-	-	-	✓	✓	✓	17.2
-	-	-	-	✓	✓	17.2
-	-	-	-	-	✓	16.4

Table 7: Performance of fine-tuning all model parameters on the test set of En-Zh style customization.

Method	BLEU	PPL	Class.
Transformer	13.7	459.6	18.0
Fine-tuning	22.3	255.5	51.2
+ Mem.-Aug. Adapter	23.2	250.6	50.4

8. Conclusion

In this work, we propose a memory-based adapter to build pluggable NMT models, which can let the users customize the generation behavior of NMT models by simply providing some text samples. We improve both the memory design and utilization to help existing models better adapt to the user-demanded styles or domains. Experiments demonstrate the superiority of our proposed method over several representative baselines. By changing the memory format, we believe our method can be applied to some other sequence generation tasks.

Acknowledgements

This work is supported by the National Key R&D Program of China (2022ZD0160502) and the National Natural Science Foundation of China (No. 61925601, 62276152, 62236011). We are also grateful to Ziang Liu and Yourun Lin for their dedicated efforts in the human evaluation process, which greatly enhanced the quality this study.

9. Bibliographical References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of ACL 2020*, pages 7747–7763.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR 2015*.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of EMNLP-IJCNLP 2019*, pages 1538–1548.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the 5th Conference on Machine Translation*, pages 1–55.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the 9th Workshop on Statistical Machine Translation*, pages 12–58.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of ICML 2022*, pages 2206–2240.

Chi Chen, Maosong Sun, and Yang Liu. 2021. Mask-align: Self-supervised neural word alignment. In *Proceedings of ACL-IJCNLP 2021*, pages 4781–4791.

Wenhu Chen, Pat Verga, Michiel de Jong, John Wieting, and William Cohen. 2022. Augmenting pre-trained language models with qa-memory for open-domain question answering. *arXiv preprint arXiv:2204.04581*.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. In *Proceedings of COLING 2018*, pages 1304–1319.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of ACL 2019*, pages 2978–2988.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *Proceedings of ICLR 2020*.

Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Fei Sha, and William W. Cohen. 2022. Mention memory: incorporating textual knowledge into transformers through entity mention attention. In *Proceedings of ICLR 2022*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of NAACL 2013*, pages 644–648.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Michael Auli, and Armand Joulin. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021a. Efficient nearest neighbor language models. In *Proceedings of EMNLP 2021*, pages 5703–5714.

Qiuxiang He, Guoping Huang, Qu Cui, Li Li, and Lemao Liu. 2021b. Fast and accurate neural machine translation with translation memory. In *Proceedings of ACL-IJCNLP 2021*, pages 3170–3180.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of EMNLP-IJCNLP 2019*, pages 2733–2743.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of ICML 2019*, pages 2790–2799.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022a. LoRA: Low-rank adaptation of large language models. In *Proceedings of ICLR 2022*.

Junjie Hu, Mengzhou Xia, Graham Neubig, and Jaime Carbonell. 2019. Domain adaptation of neural machine translation by lexicon induction. In *Proceedings of ACL 2019*, pages 2989–3001.

Zhiqiang Hu, Roy Ka-Wei Lee, Charu C Aggarwal, and Aston Zhang. 2022b. Text style transfer: A review and experimental evaluation. *ACM SIGKDD Explorations Newsletter*, 24(1):14–45.

Yongcheng Jing, Yining Mao, Yiding Yang, Yibing Zhan, Mingli Song, Xinchao Wang, and Dacheng Tao. 2022. Learning graph neural networks for image style transfer. In *ECCV 2022*, pages 111–128. Springer.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7:535–547.

Nishant Kambhatla, Logan Born, and Anoop Sarkar. 2022. CipherDAug: Ciphertext based data augmentation for neural machine translation. In *Proceedings of ACL 2022*, pages 201–218.

Pei Ke, Fei Huang, Minlie Huang, and Xiaoyan Zhu. 2019. ARAML: A stable adversarial training framework for text generation. In *Proceedings of EMNLP-IJCNLP 2019*, pages 4271–4281.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *Proceedings of ICLR 2021*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *Proceedings of ICLR 2020*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP 2014*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Tom Kocmi, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Thamme Gowda, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Rebecca Knowles, Philipp Koehn, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Michal Novák, Martin Popel, Maja Popović, and Mariya Shmatova. 2022. Findings of the 2022 conference on machine translation (WMT22). In *Proceedings of the 7th Conference on Machine Translation*, pages 1–45.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances of NeurIPS 2020*, pages 9459–9474.

Bei Li, Tong Zheng, Yi Jing, Chengbo Jiao, Tong Xiao, and Jingbo Zhu. 2022. Learning multiscale transformer models for sequence generation. In *Proceedings of ICML 2022*, pages 13225–13241.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of NAACL 2018*, pages 1865–1874.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of ACL 2021*, pages 4582–4597.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of ACL-IJCNLP 2021*, pages 6691–6706.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the ACL*, 8:726–742.

Minh-Thang Luong and Christopher Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the 12th IWSLT: Evaluation Campaign*, pages 76–79.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. UniPELT: A unified framework for parameter-efficient language model tuning. In *Proceedings of ACL 2022*, pages 6253–6264.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Proceedings of ACL 2018*, pages 312–318.

Xing Niu and Marine Carpuat. 2020. Controlling neural machine translation formality with synthetic supervision. In *Proceedings of AAAI 2020*, pages 8568–8575.

Xing Niu, Marianna Martindale, and Marine Carpuat. 2017. A study of style in machine translation: Controlling the formality of machine translation output. In *Proceedings of EMNLP 2017*, pages 2814–2819.

Xing Niu, Sudha Rao, and Marine Carpuat. 2018. Multi-task neural models for translating between styles within and across languages. In *Proceedings of COLING 2018*, pages 1008–1021.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

OpenAI. 2022. Introducing ChatGPT. (Accessed on Jun 18, 2023).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, pages 311–318.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of EMNLP-IJCNLP*, pages 2463–2473.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of EACL 2021*, pages 487–503.

Matt Post. 2018. A call for clarity in reporting bleu scores. In *Proceedings of the 3rd Conference on Machine Translation: Research Papers*, pages 186–191.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of EMNLP 2021*, pages 7930–7946.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2023. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of CVPR 2023*, pages 22500–22510.

Hrituraj Singh, Gaurav Verma, Aparna Garimella, and Balaji Vasan Srinivasan. 2021. DRAG: Director-generator language modelling framework for non-parallel author stylized rewriting. In *Proceedings of EACL 2021*, pages 863–873.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Bakhtiyar Syed, Gaurav Verma, Balaji Vasan Srinivasan, Anandhavelu Natarajan, and Vasudeva

Varma. 2020. Adapting language models for non-parallel author-stylized rewriting. In *Proceedings of AAAI 2020*, pages 9008–9015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances of NeurIPS 2017*, pages 5998–6008.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of EMNLP-IJCNLP 2019*, pages 4396–4406.

Thuy Vu and Alessandro Moschitti. 2021. Machine translation customization via automatic training data selection from the web. *CoRR*, abs/2102.10243.

Shuo Wang, Zhaopeng Tu, Zhixing Tan, Shuming Shi, Maosong Sun, and Yang Liu. 2021. On the language coverage bias for neural machine translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4778–4790.

Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. 2021a. R-drop: Regularized dropout for neural networks. In *Advances of NeurIPS 2021*, pages 10890–10905.

Xuanxuan Wu, Jian Liu, Xinjie Li, Jinan Xu, Yufeng Chen, Yujie Zhang, and Hui Huang. 2021b. Improving stylized neural machine translation with iterative dual knowledge transfer. In *Proceedings of IJCAI 2021*, pages 3971–3977.

Yuxiang Wu, Yu Zhao, Baotian Hu, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2022. An efficient memory-augmented transformer for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2210.16773*.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of NAACL 2021*, pages 3511–3535.

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. 2021. Adaptive semiparametric language models. *Transactions of the Association for Computational Linguistics*, 9:362–373.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. Improving the transformer translation model with document-level context. In *Proceedings of EMNLP 2018*, pages 533–542.

Peng Zhang, Zhengqing Guan, Baoxi Liu, Xianghua Ding, Tun Lu, Hansu Gu, and Ning Gu. 2022. Building user-oriented personalized machine translator based on user-generated textual content. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW2):1–26.

Xin Zheng, Zhirui Zhang, Shujian Huang, Boxing Chen, Jun Xie, Weihua Luo, and Jiajun Chen. 2021. Non-parametric unsupervised domain adaptation for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4234–4241.

10. Appendix

10.1. Training Details

NMT Model Training We use WMT14 (Bojar et al., 2014) De-En and WMT20 (Barrault et al., 2020) Zh-En training data to train NMT models. For all the involved language pairs (i.e., En-Zh, Zh-En, and De-En), we train the Transformer model using the following hyper-parameters. All the models are optimized by Adam (Kingma and Ba, 2014), with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We train each model for 200K iterations on 4 NVIDIA A100 GPUs, where the training speed is 8.5 iterations per second. We use the learning schedule presented in Vaswani et al. (2017), with a maximum learning rate of 7e-4 and the warm-up step is 4K. Each mini-batch contains 32K tokens in total. Both the dropout rate and the label smoothing penalty are set to 0.1. During inference, the beam size is 4. For En-Zh and Zh-En, the NMT models have 253.9M parameters. For De-En, the model has 198.3M parameters.

Adapter Training We train the proposed memory-augmented adapter for 20K iterations. The maximum learning rate is set to 2e-4 and we restart the learning rate schedule when training adapters. Each mini-batch contains 8K tokens. Each experiment is conducted through a single run. We tune the values of the hyperparameters on the validation set through grid search.

kNN Decoding To apply kNN decoding to our method, we should firstly build a datastore in the same way as that illustrated in Khandelwal et al. (2021) using our model augmented with the proposed adapters. We use the open-sourced implementation of kNN decoding.[6]

[6] https://github.com/urvashik/knnmt

10.2. Details on Baselines

In this subsection, we provide the essential details of the baselines in this work:

- *Adapter* (Houlsby et al., 2019): we use the same adapter architecture as presented in Houlsby et al. (2019). The training hyperparameters are the same as our method, excluding some newly introduced hyperparameters (e.g, α and β). The default dimension of the hidden layer is set to 64, following Houlsby et al. (2019). Since our method use more parameters than Adapter, we also train a larger adapter to assimilate the parameter count, whose hidden dimension is set to 512. The bigger adapter still performs worse than our method (16.9 vs. 20.8 in terms of BLEU), indicating that our performance improvement is not totally caused by the larger adapter size.

- *MT+Rewrite* (Syed et al., 2020): we train a rewriting model to refine the output of the NMT model. Specifically, we fine-tune a pretrained encoder-decoder model to transfer the model outputs into stylized texts.

- *kNN-MT* (Khandelwal et al., 2021): there are mainly three hyperparameters that have a significant impact on performance: k, T, and λ. We tune the hyperparameters on the validation set. For style customization, k = 128, T = 30 and λ = 0.7 in En-Zh. In Zh-En, k = 16, T = 40 and λ = 0.6. For domain customization, k = 16 across all the four domains. T = 4 in IT, Medical, and Law. T = 40 in Koran. λ is tuned to be 0.2, 0.3, 0.3, 0.6 in IT, Medical, Law, and Koran, respectively.

- *DExperts* (Liu et al., 2021): we should learn two independent language models, of which one language model serves as an expert and another one is an anti-expert. The expert model is fine-tuned on the user-provided data while the anti-expert is trained on the general domain data. α is tuned on the validation set and the final value is 0.2.

10.3. Case study

We place some translation examples in Table 8 to provide a better understanding of the difference between the involved methods. There are 6 sentences from short to long. We can find that our method always outputs better translations. Also, kNN-MT is not always better than Adapter, see the 3-rd and 4-th cases.

Syed et al. (2020) believe that the author-style can be understood at three levels, from punctuation, word usage to syntax (Syed et al., 2020). In our cases, we can find that our method can learn to generate author-style better in different granularities. From case 2, our method correctly translates the phrase “build the tower” to “造塔” while the other methods translate it to “修建塔” or “建造雷峰塔”. Although the meaning is the same, our translation is closer to the expression style of the original author/user. Similarly, our method properly translates the word “call” to “呼唤” while the other methods translate it to “打电话” or “叫” in case 3. Also, our method translates the phrase “that night” to “那夜” while the other methods translate it to “那一晚” or “昨夜” in case 4. Furthermore, it can also be easily found that our method can generate similar sentences that have similar syntactic styles. From case 1 and case 6, we can see that the sentence generated by our method is more similar to the reference in terms of sentence segmentation.

These cases from lexical level to syntactic structure also demonstrate the rationality and effectiveness of our multi-granularity memory design.

10.4. Application to Larger Model

We also conduct experiments on a model of a larger scale, whose hidden size is 1024 and parameter size is 596.0M. On the test set of En-Zh style customization, our memory-augmented adapter (without kNN decoding) outperforms Adapter (Houlsby et al., 2019) by 2.9 BLEU and kNN-MT (Khandelwal et al., 2021) by 2.2 BLEU. This demonstrates that our method is also effective when the model size is larger. How to apply our method to larger models deserves further exploration.

Table 8: Case study on En-Zh style customization. For clarity, we only choose the representative baselines (i.e., Adapter (Houlsby et al., 2019) and kNN-MT (Khandelwal et al., 2021)) for comparison.

Source	So biological truth is by no means a talisman for polygamy.
Adapter	因此，生物学真理决不是一夫多妻制的护身符。
k**NN-MT**	所以，生物学的真理，决不是一夫多妻制的护身的挡牌。
Ours	所以生物学上的道理，决不是一夫多妻的护身符。
Reference	所以生物学的真理，决非多妻主义的护符。
Source	Could it be that when he built the tower, he didn't think that the tower would fall down after all.
Adapter	难道他修建塔的时候，总不认为塔会倒塌吗。
k**NN-MT**	倘若他建造雷峰塔的时候，他没有想到那塔终究会倒塌。
Ours	难道他造塔的时候，总不觉得那塔到底要倒下去么。
Reference	莫非他造塔的时候，竟没有想到塔是终究要倒的么。
Source	It was the morning of the fifth day that everyone dragged him up early in the morning and stood on the shore listening to the call.
Adapter	这是第五天早晨，大家一大早就把他拖起来，站在岸上听着叫。
k**NN-MT**	第二天的早晨，大家一大早把他拖起来，站在岸上听听差打电话。
Ours	这是第五天的早晨，大家早把他拖起来，站在岸上听着呼唤。
Reference	就是这第五天的早晨，大家一早就把他拖起来，站在岸上听呼唤。
Source	Really, until now, I really haven't eaten the good beans like that night, and I haven't seen the good show like that night anymore.
Adapter	真的，到目前为止，我实在没有吃过那一晚这样的好豆子，我再也没有见过那晚这样的好秀了。
k**NN-MT**	真的，直到现在，我实在没有吃过昨夜的豆子，我也从来没见过这样的好东西。
Ours	真的，直到现在，我实在没有吃过那夜那样好的豆子了，我也再没有见过那夜那样的好节目了。
Reference	真的，一直到现在，我实在再没有吃到那夜似的好豆，也不再看到那夜似的好戏了。
Source	This affair happened at the turn of winter and spring. The wind was not so cold anymore, and I wandered outside for a longer time; by the time I got home, it had been probably already dark.
Adapter	这事发生在冬春交替的时候，风不再那么冷了，我在外面漫游了更长的时间；到我到家的时候，大概是已经暗了。
k**NN-MT**	这事发生在冬或今年春末，微风不再那么冷，我徘徊了大半天，到我回家的时候，天气大概已经很深了。
Ours	这事发生于冬春之交，风已不再那么冷，我在外面徘徊了更长的时间；到我到家的时候，大概已经天黑了。
Reference	这是冬春之交的事，风已没有这么冷，我也更久地在外面徘徊；待到回家，大概已经昏黑。
Source	For example, to build a railway, if we tell them how beneficial this thing is, they will never listen. If we, according to a myth, tell that previously a great immortal pushed a wheelbarrow over the rainbow, and now we imitate him to build a road, then everything can be done.
Adapter	举例来说，要修建铁路，如果我们告诉他们这事是何等的益处，他们就决不肯听从，如果我们，根据一个神话，告诉以前一个伟大的不朽的推车推着彩虹之上，现在我们模仿他修筑一条路，那么一切都可以完成了。
k**NN-MT**	譬如说，如果我们告诉他们，铁路的建设是多么有益，他们决不听，如果我们根据神话说，先前一个大不朽的独轮车从彩虹上推过，现在我们模仿他造路，便可以做点事了。
Ours	譬如造铁路罢，倘告诉他们这东西有多大益处，他们便决不肯听话，倘使我们据传说，说先前一个伟大的不朽的车手推着彩虹，现在就模仿他来造一条路，那么，一切便都可以做。
Reference	譬如要造一条铁路，倘若对他们说这事如何有益，他们决不肯听；我们如果根据神话，说从前某某大仙，曾推着独轮车在虹霓上走，现在要仿他造一条路，那便无所不可了。